그래픽 사용자 인터페이스를 이용한 병렬 프로그래밍 환경 설계 및 구현

이원용[†] · 박두순[††]

요 약

본 논문은 그래픽 사용자 인터페이스를 이용하여 병렬 프로그래밍 환경을 설계하고 구현하였다. 병렬 프로그램은 다양한 하드웨어의 특성에 따라 또는 프로그램의 특성에 따라 사용자가 병렬 프로그램을 작성하여야 하기 때문에 사용자가 병렬 프로그램을 작성하는 것은 매우 어렵다. 본 논문에서는 이런 문제를 도와주기 위하여 기존의 병렬 컴파일러에서 제공되고 있는 텍스트 위주의 병렬화 정보 대신에 그래픽 사용자 인터페이스를 이용하여 편안하고 쉽게 병렬화 정보를 제공하는 병렬 프로그래밍 환경을 제안하고, 구현하였다. 본 논문의 병렬 프로그래밍 환경은 종속성 분석, CFG, HTG, 루프 병렬화 등의 기능을 제공한다.

A Design and Implementation of Parallel Programming Environment using Graphical User Interface

Won-Yong Lee[†] and Doo-Soon Park[††]

ABSTRACT

In this paper, we design and implement a parallel programming environment using graphicial user interface. Parallel program must be written according to the characters of various hardwares or programs. So it is difficult to write the parallel program. In this paper, we design and implement a parallel programming environment which solved this problem. The traditional parallel compiler provides the parallel information in the text environment. But this paper provides the parallel information using graphicial user interface so that the user may use it more easily. This parallel environment provide functions such as, data dependence analysis, CFG, HTG, and loop transformation.

1. 서 론

정보 산업의 발전에 따라 컴퓨터의 응용분야는 점점 방대해 지고 있다. 이에 따라 방대한 양의 정보를 빠르게 처리할 수 있도록 많은 연구가 진행되고 있으며 이 연구 중의 하나로 병렬 컴퓨터가 등장하게 되었다. 그러나 병렬 컴퓨터는 사용하는 과정에서 많은 어려움이 발생되고 있다. 즉 개발 환경에 따라 병렬 프로그래밍 환경이 서로 호환되지 않으며 처음 사용하는 사용자는 다양한 병렬 컴파일러 구조, 병렬 운영체제를 이해하여야 하는 등 많은 어려움이 있다. 따라서 이런 문제를 해결하기 위해 병렬 프로그램을 쉽게 작성할 수 있도록 도와주는 병렬 프로그래밍 환경에 대한 여러 방법들이 연구되고 있다[1,2].

본 논문에서는 그래프 중간 표현 형태를 기반으로 한 병렬 프로그래밍 환경을 설계, 구현하였으며 메뉴 선택방식을 채택하여 처음 사용하는 사용자도 즉시 병렬 프로그래밍 환경에서 작업 할 수 있도록 설계하였다.

이를 위하여 순차 프로그램을 입력으로 받아 종속성을 분석하여 문장 간의 종속성 정보를 제공한다. 그리고 병렬 프로그래밍 환경에서 기본적으로 사용되는 제어 흐름과 종속성 정보를 분석하여 중간 그래

† 정회원, 혜전대학 컴퓨터 계열 부교수
†† 정회원, 순천향대학교 정보기술공학부 교수

프인 제어 흐름 그래프(CFG)와 함수형 병렬성을 유도하는 계층적 태스크 그래프(HTG)를 사용자에게 보여준다. 또한 종속성 정보를 이용하여 루프 문장의 교환(Interchange), 분산(Distribution), 융합(Fusion)의 가능성을 판별하고, 각각의 방법에 따라 변환 후의 프로그램을 보여준다.

따라서 사용자는 이 병렬 프로그래밍 환경에서 제공된 정보를 이용하여 중간 코드의 변환 및 프로그램의 병렬화, 최적화에 필요한 최적의 정보를 이용할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 그래프 중간 표현 형태의 연구동향을 살펴본다. 3장에서는 본 논문에서 제시한 병렬 프로그래밍 환경의 구현방법을 기술하고 4장에서는 병렬 프로그래밍 환경을 구현하여 사용자에게 구현된 정보를 보여준다. 그리고 5장에서 결론을 맺는다.

2. 그래프 중간 표현 형태

병렬 환경에서 좀 더 개선된 병렬 프로그램을 얻기 위해 많은 연구가 진행되고 있다. 그 중에서 그래프 중간 표현 형태의 연구는 내부적으로 변환과정의 정보 상태를 유지하면서 표현되는데, 이렇게 그래프 형태로 프로그램을 유지하면서 벡터화나 병렬화에 이용되는 그래프 중간표현 형태는 CFG, PDG(Program Dependence Graph), HTG 등이 있다[3].

CFG는 명령의 제어흐름을 나타내며 제어 종속성 정보를 표현하는 중간 그래프 형태이다[4]. PDG는 Ferrante에 의해 제시된 중간 그래프로 벡터화 또는 병렬화에 용이한 프로그램 중간표현 형태로 제어 종속정보, 자료 종속정보를 한 그래프에 표현하였다[5]. HTG는 Girkar에 의해 제안되었는데 병렬 프로그램의 중간 그래프 형태로 자료 종속성과 제어 종속성을 그래프 상에서 최소화하여 표현한다. 또한 태스크 단위 또는 함수단위의 병렬성을 추출하고 표현한다[6].

3. 병렬 프로그래밍 구현 방법

본 논문에서 구현된 병렬 프로그래밍 환경 중 종속성 분석, CFG, HTG 및 루프 변환에 대한 구현 방법을 서술한다[4,6,7].

3.1 종속성 분석 구현 방법

종속성 분석은 자료 흐름분석과 제어 흐름분석을 통하여 그 정보를 이용하여 종속성을 분석한다. 본 논문에서는 기존에 제안된 알고리즘인 GCD 테스트와 Banerjee 부등식을 이용하였다[8,9].

본 논문의 종속성 분석은 순차 프로그램을 입력으로 받아 종속성 분석 정보를 출력한다. 종속성 분석의 목적은 프로그램 내에 존재하는 모든 루프에 대하여 루프내부에 포함되는 문장들의 모든 쌍이 서로 종속 관계에 있는가를 테스트하기 위한 것이다.

종속성 분석 방법은 첫째 종속성 테스트 및 문장 비교 프로그램으로 이 프로그램 내에서는 루프내부의 모든 문자 쌍에 대하여 루프의 각 레벨에서 종속 관계를 비교한다. 두 번째 단계는 종속성을 결정하는 단계로 임의의 루프 레벨에서 선정된 임의의 문장 쌍은 종속 발생을 유발시키는 변수와 변수에서 사용되는 첨자가 존재하는지를 발견하여 존재할 경우에 GCD 테스트와 Banerjee 부등식을 적용하여 종속성 여부를 결정한다. 이러한 종속성 분석하는 방법은 그림 1에서 보여주고 있다.

```
for 종속성이 존재 가능한 각 문장 쌍에 대하여 do
  while 2개의 문장에 있는 배열 A에 대한 테스트할
                                    참조가 존재 do
     for 문장 쌍의 종속 가능한 각 루프 level L에 대하여 do
        독립적이면 ← NO;
         while 독립적이 아니면 and
         배열 A의 테스트할 첨자가 존재 do
          if 배열 A의 첨자식이 linear then
            if gcd test show independece or
                 Banerjee Inequality does't hold then
                 INDEPENDENT ← TRUE;
          endwhile;
             if 독립적이 아니면 then
                Record the level L dependence;
     endfor;
  endwhile;
endfor;
```

그림 1. 종속성 분석 방법

3.2 CFG

CFG는 제어흐름을 나타내는 방향성 그래프로 G=(V, E) 형태로 표현된다. 여기서 V는 노드를 나타내고, 노드는 프로그램의 기본 블록이나 한 문장 단위를 나타낸다. E는 연결선을 나타내는데 두 노드 사이의 제어흐름을 나타낸다. CFG는 한 개의 START 노드와 STOP 노드를 갖는다. CFG는 제어 종속 관

계를 나타내기 위해 포스트 도미네이터(post-dominator) 개념을 이용하여 제어 종속성을 계산한다[4].

CFG는 기본 블록에서 유도되는데, 루프의 존재여부에 따라 2가지 방법으로 구분된다. 첫째 루프가 포함되지 않은 기본 블록은 제어 흐름에 따라 연결선을 연결하면 CFG 구성할 수 있다. 둘째 루프가 포함된 기본 블록은 루프 입력부와 루프 몸체, 루프 출구부의 반복 과정을 거쳐 CFG를 구성 할 수 있다.

루프 입력부는 외부로부터 제어 흐름이 들어와 초기 수행에 적당한가를 검사하는데 루프 수행조건을 검사하여 정당하지 않은 경우 루프 출구부로 분기하여 작업 종료 처리 후 빠져나간다. 그러나 정상적인 경우 루프 몸체부로 들어가 한번 반복 수행 후 출구부에서 루프 특성에 따라 조건변수 변환작업 후 조건을 검사하고 다음반복을 수행하거나 루프 출구부 작업 종료 후 루프영역을 벗어난다. 그림 2는 예제 프로그램과 CFG를 나타내고 있다.

(a) 예제 프로그램 (b) 제어흐름 그래프(CFG)

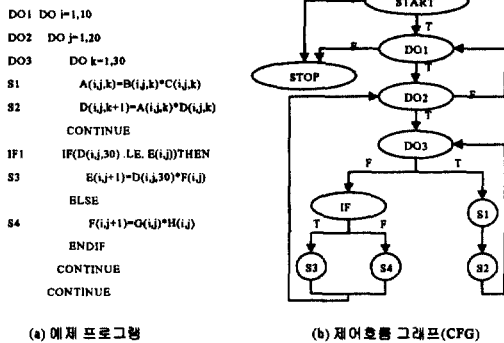



그림 2. 예제 프로그램과 CFG

3.3 HTG

HTG는 계층적 태스크 개념을 도입하고 자료 제어간의 종속성을 제거하여 중간 그래프를 완성한다[6].

HTG는 CFG로부터 유도되는데, 유도 과정은 CFG를 DFS(Depth first search)를 수행하여 후위경로(back arc)를 추출하여 그와 관계된 경로를 계층화하여 HTG를 구축한다. 그림 4는 그림 3의 CFG를 DFS를 수행한 결과를 나타낸 것이다.

수행 결과 4가지 경로를 발견할 수 있다.

1) 트리 경로는 노드 y가 방문치 않는 경로 (x,y) 가리키며 {(1,2),(3,4),(4,5),(4,6),(5,7),(7,8),(8,9),8,10)} 로 표시된다.

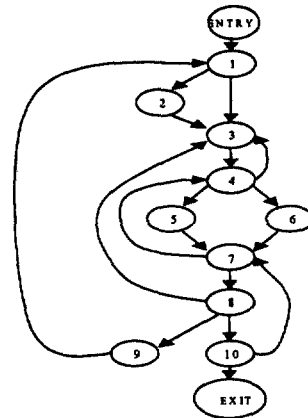



그림 3. CFG

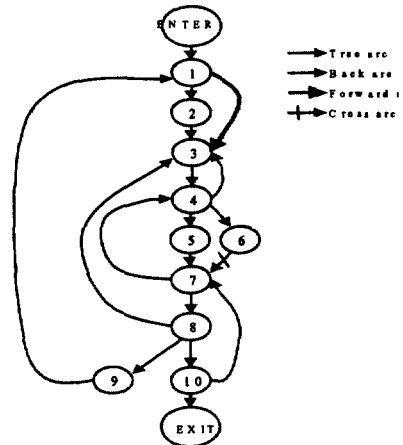



그림 4. CFG의 DFS 수행 결과

2) 후위 경로는 트리로 구성된 y로부터 x의 경로가 하나 존재하는 경우를 말하며 이 경로를 트리 경로라 하며, y는 DFS 트리 경로에서 x의 선행자라고 하며 {(9,1),(8,3),(7,4),(4,3),(10,7)}로 표시 할 수 있다.

3) 전위 경로는 노드 y는 DFS 트리에서 x의 후행자인 경우로 (x,y)는 전위경로라 하며 {(1,3)}으로 표시된다.

4) 교차 경로는 y는 방문했지만 x의 선행자도 후행자도 아닌 경우로 이 경우 (x,y)는 교차경로라 하며 이는 {(6,7)}로 표시할 수 있다. 여기서 경로c(x,y)에 대해서 x는 "source"가 되고 y는 그 경로의 "sink"가 된다.

다음으로 후위경로 갖는 노드의 집합 H(G)를 구하는 과정이다. 노드의 집합 H(G)를 후위 경로에 대해 sink인 모든 노드의 집합이라고 하면, 여기서

(y,x)는 후위경로로 표시할 수 있고 이 H(G)는 H(G) ={1,3,4,7}로 표시할 수 있다.

다음은 H(G)에 대하여 루프를 표시하고 정의하는 과정이다. T(x)는 DFS 경로에서 x의 후행자를 나타내고 있다고 가정하면 이 경우 후위 경로에 대한 루프는 표 1로 표시 할 수 있다.

표 1. 후위 경로 루프 표현

back arc	loop
L(9,1)	{1,2,3,4,5,6,7,8,9,10}
L(8,3)	{3,4,5,6,7,8,10}
L(7,4)	{4,5,6,7,8,10}
L(10,7)	{7,8,10}
L(4,3)	{3,4,5,6,7,8,10}

다음으로 H(G)내에서 모든 노드 x에 대해서 강하게 연결 돼 있는 영역 I(x)를 정의하면 표 2와 같이 나타낼 수 있다.

표 2. CFG 연결 관계 표현

A	선행자 집합 P(A)
I(1)	{V}
I(3)	{I(1),V}
I(4)	{I(3),I(1),V}
I(7)	{I(1),I(3),I(4),V}
V	{ }

마지막으로 루프 멤버들의 선행관계 정의하여 계층화하는 단계이다. 표 3은 멤버들의 선행관계를 나타낸 것이다.

표 3. 멤버들의 선행 관계

Node x	Region I(x)
I(1)	{1,2,3,4,5,6,7,8,9,10}
I(3)	{3,4,5,6,7,8,10}
I(4)	{4,5,6,7,8,10}
I(7)	{7,8,10}

수직 계층성을 이용하여 각 노드를 계층화 한 HTG가 그림 5에서 보여 주고 있다.

3.4 루프 변환

루프의 반복작업은 적은 문장의 수행에도 반복작업을 하기 때문에 작업 속도에 많은 영향을 미친다.

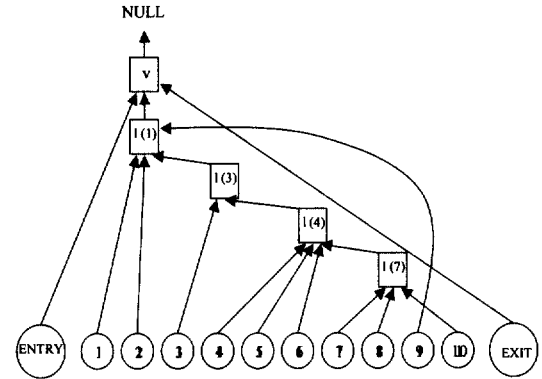



그림 5. HTG

따라서 효율적인 수행을 하기 위해서 루프 병렬화는 병렬 프로그래밍 환경에서 중요하다. 본 절에서는 루프 변환 중 루프교환, 루프융합 및 루프분산에 대한 구현방법을 기술한다[7,10].

루프 교환을 구현하기 위해서는 각각의 루프 인덱스 역할을 하는 숫자를 부여하고 그리고 종속성 정보를 나타내는 방향벡터 중에서 각각의 루프 인덱스에 해당되는 부분을 찾아내 루프 교환여부가 가능한지 판단한다. 그리고 전체의 루프에서 교환할 수 있는 루프의 인덱스 쌍을 조합하여 보관하고, 전체 집합에서 루프교환 가능조건에 만족하는 경우만 인덱스 쌍으로 지정하여 교환한다.

루프 융합은 이웃하고 있는 두 루프를 하나의 루프로 변환하는 것으로 루프 변수의 영역이 동일해야 하며 융합했을 때 자료 종속성 정보가 변하지 말아야 한다. 그러나 융합을 해서 자료 종속정보가 유지되지 못할 경우는 루프 융합을 할 수 없다. 이런 루프 융합의 타당성 검증은 먼저 이웃하는 두 개의 루프를 하나의 루프로 변환한 뒤 문장 간의 종속성 여부를 검증하여 역 종속이 존재하면 루프융합이 불가능하다.

루프 분산은 하나의 루프 안에 있는 여러 개의 문장들을 자료 종속관계에 따라 좀더 작은 단위로 분할하는 방법이다. 나뉘어진 새로운 루프들은 원래의 프로그램 문장들을 나누어 갖게되고 원래의 루프와 같은 회전 범위를 가지고 수행한다.

4. 병렬 프로그래밍 환경 구현

본 장에서는 그래픽 사용자 인터페이스를 이용하여 설계한 병렬 프로그래밍 환경에 대하여 소개하고

그 기능 및 작동방법에 대하여 기술한다.

4.1 병렬 프로그래밍 환경 특징 및 기능

본 논문에서 구현한 그래픽 사용자 인터페이스를 기반으로 한 병렬 프로그래밍 환경은 다음과 같은 특징을 가지고 있다.

첫째 사용자가 직접 병렬 프로그래밍 환경에서 프로그램 작성이 용이하도록 그래픽 인터페이스 방식과 메뉴 선택 방식을 이용하여 화면에서 프로그램을 입력, 수정, 편집이 가능하도록 설계하였다. 둘째 개방 사용자 환경을 적용하여 특정 하드웨어와 독립적으로 작업할 수 있도록 환경을 제공한다. 즉 누구나 사용하기 쉽도록 윈도우 98 운영체제에서 C 언어로 설계 및 구현하였다. 셋째 연속적인 병렬 작업을 수행하여 변환과정을 쉽게 분석 할 수 있다. 병렬 환경 내에서 순차 프로그램을 입력하여 기본 블록, 종속성 분석, 중간 그래프를 작성할 수 있으며 루프 병렬화 작업을 수행 할 수 있다. 넷째 기존의 수작업으로 작성하던 그래프를 직접 출력할 수 있으며, 도움말 기능을 제공하여 처음 사용하는 사용자도 병렬환경을 쉽게 접할 수 있다. 그림 6은 본 논문에서 구현한 병렬 프로그래밍 환경을 보여주고 있다.

그림 6. 병렬 프로그래밍 환경

4.2 구현 환경

본 논문에서 제공하는 병렬 프로그래밍 환경은 사용하기 쉽고, 사용자가 병렬 프로그래밍 환경에 접근이 쉽도록 X-Window 기반의 그래픽 인터페이스를 사용하였으며, 메뉴 선택방식을 적용하여 설계 및 구현하였다. 병렬 프로그래밍 환경은 네 가지 기능을 제공한다.

첫째는 사용자에게 직접 창에서 프로그램의 작성을 도와주는 파일 편집 기능이며, 둘째는 종속성 정보를 분석하여 사용자에게 종속성 정보를 제공하는 기능, 셋째 그래프 중간 표현 형태를 제공하는 자동 병렬 변환 기능이며 마지막으로 병렬화 기능으로 루프의 변환 정보를 사용자에게 제공한다.

1) 파일 및 편집

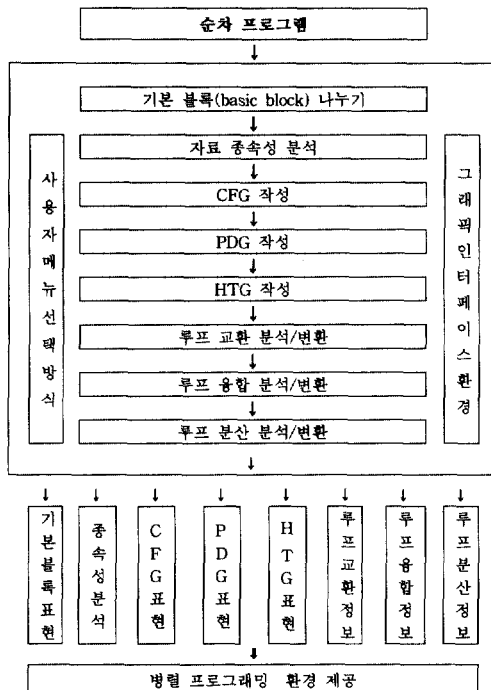
파일 및 편집기능은 사용자가 직접 창에서 입력, 수정, 삭제, 조회 할 수 있고, 덧붙임을 할 수 있도록 설계하였다. 이 기능은 2가지 기능이 있는데 첫째 파일 메뉴로 이 기능은 6개의 부 메뉴로 구성되어 있으며 편집메뉴는 3개의 부 메뉴로 구성된다.

2) 종속성 분석 기능

종속성 분석 기능은 병렬 프로그래밍 환경에서 가장 문제가 많이 발생되고, 오류를 범하기 쉬운 종속성 정보를 분석하는 과정이다. 종속성 분석은 현재의 문장을 기준으로 흐름 종속, 역 종속, 출력 종속의 존재 여부를 파악하여 문장 간의 정보를 제공하는 기능이다. 병렬 프로그래밍 환경에서는 문장 간의 종속성 종류를 분석, 표현하고 종속성 관계를 분석하여 사용자에게 알아보기 쉽도록 정보를 제공한다.

3) 중간 그래프 표현 기능

CFG는 기본 블록을 제어 흐름에 따라 연결하여 CFG를 작성한다. 이 CFG는 첫 번째 기본 블록과 마지막 기본 블록은 entry와 exit 부분으로 삽입하고 이를 노드로 표현하며, 노드는 "□" 로 표시하였다.

HTG는 CFG를 이용하여 DFS를 수행하여 후위경로 집합을 추출한 뒤 수직 계층적 루프의 개념을 도입하여 중간 그래프를 작성한다. 병렬 프로그래밍 환경에서는 소스코드를 입력하여 이를 계층적 태스크

그래프 형태로 볼 수 있도록 기능을 제공한다.

4) 병렬화 기능

루프 교환은 다중루프에서 미세 단위의 병렬화나 대단위 병렬화를 위하여 루프 순서를 변경하는 것으로 병렬 프로그래밍 환경에서는 첨자의 변환과정과 변화된 작업정보를 사용자에게 제공한다.

루프 융합은 이웃하고 있는 두 루프를 하나의 루프로 변환하는 것을 말하는데 이 병렬 프로그래밍 환경에서는 두 문장의 루프 융합과정을 화면에 보여주면서 정보를 제공한다.

루프 분산은 하나의 루프 안에 있는 모든 문장들이 종속관계로 인해 병렬화를 수행할 수 없을 때 변환하는 방법으로 병렬 프로그래밍 환경에서는 종속관계를 분석하여 루프 분산 여부를 분석한 후 분산된 정보를 사용자에게 제공한다. 표 4.는 사용자 메뉴를 나타낸다.

표 4. 사용자 메뉴

작업구분	메뉴	작업 기능
파일	newfile	새프로그램 입력
	open	기존 프로그램 열기
	load	기존 프로그램 수정
	save	프로그램 저장
	save as	다른이름으로 저장
	quit	프로그램 종료
편집	cut	편집내용 일부삭제
	add	편집내용 덧붙임
	delete	프로그램 삭제
기본블록	기본블록	기본 블록 나누기
종속성	종속성 분석	종속성 정보 출력
		종속성 그래프 표현
중간 그래프	CFG	제어 흐름 그래프
	HTG	계층적 태스크 그래프
루프 변환	loop interchange	루프 교환
	loop distribution	루프 분산
	loop fusion	루프 융합
출력	화면내용 출력	
도움말	작업내용 도움말	

4.3 병렬 프로그래밍 환경의 구현

사용자는 프로그램을 수행하면 그림 7의 초기 메뉴 창이 보여진다. 병렬 프로그램을 작성하기 위하여 newfile을 선택하면 파일을 편집할 수 있는 창이 나타난다. 사용자는 이 병렬 프로그램을 작성한 후 save라는 명령단추를 선택하면 프로그램이 저장된다. 이 과정에서 사용자는 파일 편집 기능을 이용하여 작업중의 내용을 편집, 수정할 수 있다. 그림 7은 병렬 프로그래밍 환경의 초기 메뉴 창을 보여준다.

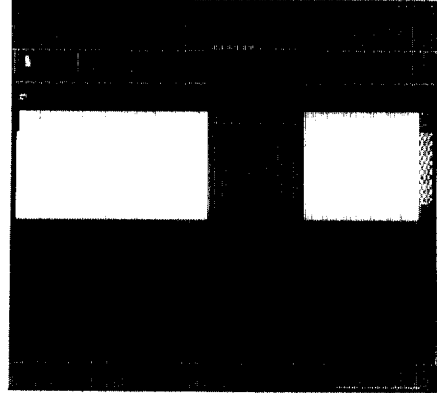



그림 7. 초기 메뉴창의 모습

순차 프로그램을 기본 블록으로 표현한 것은 그림 8에서 보여준다.

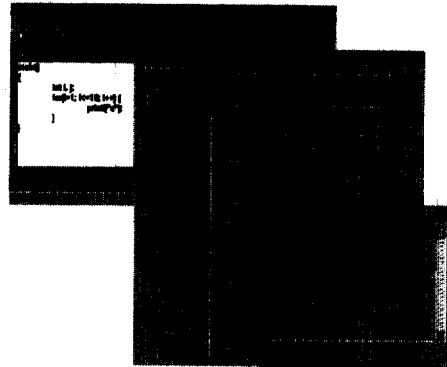



그림 8. 기본 블록 표현

종속성 분석은 특정 프로그램을 선택하여 종속성 관계를 분석하고자 할 때 사용한다. 사용자는 종속성 분석을 하기 위해서는 먼저 분석하고자 하는 프로그램을 오픈하고 종속성 분석 명령을 입력하면 프로그램 내에서 어휘분석과 구분분석을 수행한 후 자료흐름분석과 종속성 분석을 수행한다. 구현에서는 현재 계산된 값을 보관하고 프로그램이 종료된 후 변경값을 비교하여 문장간의 종속성 정보를 나타낸다. 그림 9는 프로그램 종속성 분석을 하였을 경우의 창의 모습을 보여주고 있다.

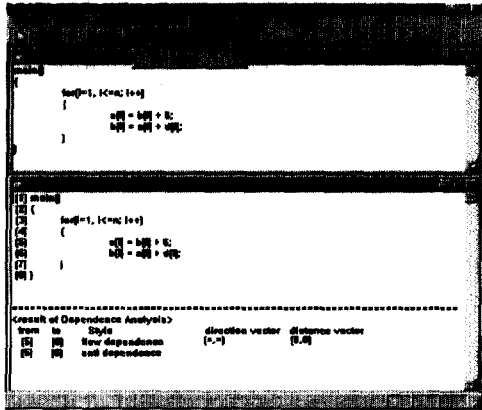



그림 9. 종속성 분석 결과

CFG는 기본 블록에서 제어 흐름을 그래프로 표현한 것이다. CFG 명령 단추를 입력하면 화면에 CFG가 작성되어 창에 보여준다. 그림 10은 순차 프로그램을 입력하면 그에 해당되는 CFG가 표현된 모습을 보여주고 있다.

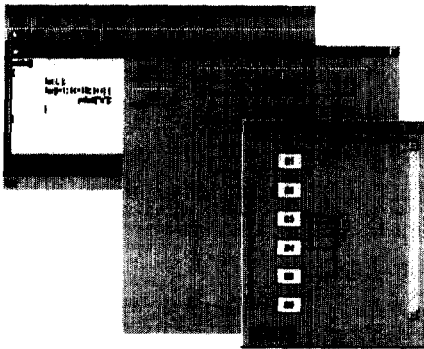



그림 10. CFG의 표현

그림 11은 HTG를 표현한 중간 그래프이다. HTG를 표현하기 위해서는 먼저 CFG를 작성한 후 이를 DFS를 수행하여 후위 경로 추출하여, 루프 멤버의 선행관계를 정의한 후 이를 계층적으로 표현한 후 HTG를 작성한다.

루프 교환은 원시 프로그램의 의미가 손상되지 않도록 루프의 수행 순서를 바꾸는 것으로, 사용자는 루프 교환 여부를 파악한 후 캐시 라인 요구수가 적은 루프를 안쪽으로 순서를 바꾼다. 그림 12는 루프 변경 전, 후의 창의 모습을 보여주고 있다.

루프융합은 이웃하고 있는 두 루프를 동일 영역으로 합치는 것으로 융합했을 때 데이터의 종속성 정보가 변하지 말아야 한다. 그림 13은 루프 융합 전, 후의 창의 모습을 보여 주고 있다.

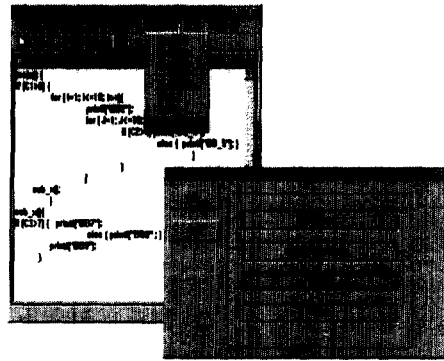
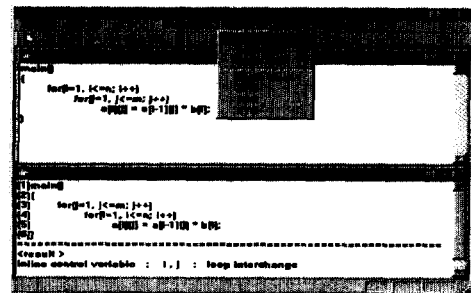



그림 11. HTG의 표현

그림 12. 루프 교환 전, 후의 창의 모습

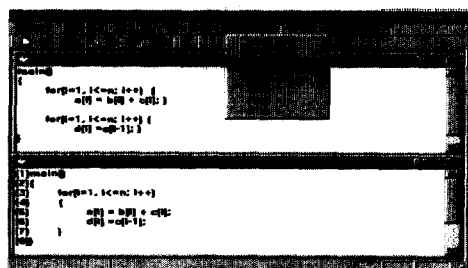



그림 13. 루프 융합 전, 후의 창의 결과

루프분산은 하나의 루프 안에 있는 여러 개의 문장들을 자료 종속관계에 따라 좀더 작은 단위로 분할하는 방법이다. 루프 분산을 수행하기 위해서는 프로그램을 메모리에 load하여 원래의 프로그램의 문장들을 나누어 갖도록 하고, 원래 루프와 같은 회전 범위를 갖도록 한다. 그러기 위해 먼저 문장 간의 종속성 정보를 분석하고 종속성이 존재하지 않을 경우 루프 분산을 수행한다. 그림 14는 루프 분산을 수행한 후의 창의 모습을 보여준다.

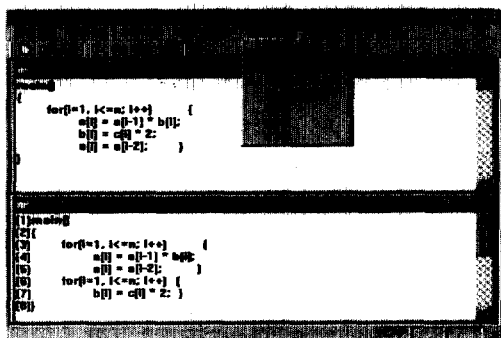



그림 14. 루프의 분산 전, 후의 창의 결과

5. 결 론

본 논문에서는 병렬 프로그래밍 환경에 대해 소개하고, 병렬 프로그래밍 환경을 제공하기 위하여 그래픽 사용자 인터페이스로 구현한 병렬 프로그래밍 환경을 제안하고 설계 및 구현하였다.

논문에서 제공하는 병렬 프로그래밍 환경은 메뉴 선택방식을 적용하여 사용자가 쉽게 사용할 수 있도록 설계 및 구현하였으며, 그래프 중간 표현 형태를 자동 병렬 변환 환경을 제공하여 원시 코드를 입력하면 즉시 중간 그래프를 제공한다. 또 특정 하드웨어와는 상관없이 작업할 수 있는 개방 사용자 환경, 종속성 분석 기능, 루프 병렬화 기능 등을 제공한다. 사용자는 병렬 프로그래밍 환경에서 작업하므로 작업시간의 단축은 물론 메모리 및 캐시 메모리를 효율적으로 이용할 수 있으며 병렬화, 최적화를 위한 각종 정보를 제공받을 수 있다. 향후 연구 방향으로는 좀더 확장된 병렬 환경을 제공하기 위해 본 연구에서 제공되지 않은 중간그래프를 더 추가하여 개발하고, 하드웨어적인 특성을 고려하여 병렬 프로그래밍 환경을 제공할 수 있도록 연구되어야 할 것이다.

참 고 문 헌

[1] Allen J. R. and K. Kennedy, "A Parallel Programming Environment," IEEE Software, Vol 2, No 4, pp 21-29, July, 1985.

[2] Allen R. and K. Kennedy, "Automatic Translation of FORTRAN program to Vector Form," ACM Transactions on Programming Languages and Systems, Vol. 9, No. 4, pp 491-542, October 1987.

[3] 박성순, "그래프 중간 표현 형태를 이용한 프로그램 벡터화," 고려대학교, 박사학위논문, 1993.

[4] Aho, A.V., R. Sethi, J.D. Ullman, "Compilers : principles, Technique, and Tools Addison-Wesley," Reading, Massachusets, 1986.

[5] Ferrante, J., K.J. Ottenstein, J. D. Warren, "The Program Dependence Graph and It's Use in Optimization", Lecture Notes in computer Science Vol. 167, Springer-Verlag, pp 125-132, 1984.

[6] Girkar, M. Baburao, "Functional Parallelism: Theoretical Foundations and Implementation" Ph. D. Thesis, U. of Illinois at Urbana-Champaign, 1992.

[7] 장유숙, "자료종속성 제거 방법을 이용한 프로시져 변환에 관한 연구" 박사학위 논문, 순천향대학교, June, 2001.

[8] Banerjee U., "Dependence Analysis for Supercomputing", Kluwer Academic publishers, Boston, MA, 1988.

[9] Allen J. R. "Dependence Analysis for Subscripted variables and its Applications to Program Translation", Ph. D Thesis, Dept. of Mathematical Science, Rice University, Houston Texas, April, 1983.

[10] 이만호, "병렬화를 위한 루프의 구조의 변환", 정보과학회지 제 12권 제 5호, pp 54-56, June, 1994.

이 원 용

1980년 중앙대학교 공과대학 전자계산학과 졸업(공학사)
1985년 중앙대학교 국제경영대학 경영정보학과 졸업(경영학석사)
1996년~현재 순천향대학교 대학원 전산학과 박사과정
1993년 9월~현재 혜전대학 컴퓨터계열 부교수
관심분야 : 병렬처리, 컴파일러, 프로그램언어

박 두 순

1981년 고려대학교 수학과 졸업(이학사)
1983년 충남대학교 대학원 전산학과 졸업(이학석사)
1888년 고려대학교 대학원 전산학 전공(이학박사)
1992년~1993년 미국 U. of Illinois at Urbana-Champaign CSRD 객원교수
2000년~현재 순천향대학교 컴퓨터교육원 원장
1985년~현재 순천향대학교 정보기술공학부 교수
관심분야 : 병렬처리, 컴파일러, 멀티미디어 정보검색, 가용성, 컴퓨터교육